ELSEVIER Applied Mathematics and Computation 92 (1998) 195–218

APPLIED MATHEMATICS AND COMPUTATION

An efficient approach for the numerical simulation of multibody systems

R. Sudarsan [a,*], S. Sathiya Keerthi [b,1]

[a] *Department of Engineering Management, George Washington University, Washington, DC 20052, USA*

[b] *Department of Computer Science and Automation, Indian Institute of Science, Bangalore-560 012, India*

Abstract

The field of kinematics and dynamics of mechanical systems has progressed from a manual graphics art to a highly developed discipline in analytical geometry and dynamics. Various general purpose formulations for the dynamic analysis of Constrained Mechanical Systems (CMS) lead to mixed Differential-Algebraic Equations (DAEs) called the Euler–Lagrange equations. During the past fifteen years many contributions have been made to the theory of computational kinematics and dynamics of CMS (also called Mutibody dynamics). The recent advances in computer hardware and software have tremendously revolutionized the analysis of CMS. There are various numerical approaches for solving general vector fields. In the previous paper [Appl. Math. Comput. 92 (1998) 153–193] a complete and detailed analysis of various approaches for the numerical solution of vector fields is given. In this paper we extend the algorithms presented in the above mentioned reference to solve the Euler–Lagrange equations of motion for CMS. The numerical experiments suggest that perturbation approach performs 'better' than the other approaches.

AMS classification: 65L05; 65L06; 70H35

Keywords: Vector fields; Differential-algebraic equations; Manifolds; Local parameterization; Constraint stabilization; Euler–Lagrange equations; Multibody systems; Numerical ODEs

* Corresponding author. E-mail: sudarsan@cme.nist.gov.

[1] E-mail: ssk@csa.iisc.ernet.in.

0096-3003/98/$19.00
PII: S0096-3003(97)10041-8

1. Introduction

The field of kinematics and dynamics of mechanical systems has progressed from a manual graphics art to a highly developed discipline in analytical geometry and dynamics. Various general purpose formulations for the dynamic analysis of constrained Mechanical Systems (CMS) lead to mixed Differential-Algebraic Equations (DAEs) called the Euler–Lagrange equations. During the past 15 years many contributions have been made to the theory of computational kinematics and dynamics of CMS (also called Mutibody dynamics). The recent advances in computer hardware and software have tremendously revolutionized the analysis of CMS. The status and development of multibody dynamics is documented in Refs. [1,2] and also in a recent special issue on DAEs [3]. While in the past, engineers dealt with small scale systems that could be analyzed by clever analytical formulations, general purpose codes now permit the design and analysis of complex systems. As the complexity of the systems increases, so does the need for fast and reliable numerical procedures for solving the equations of motion. The numerical approaches for solving general vector fields are: (1) Parameterization approach (Tangential Parameterization (TP)); (2) Constraint Stabilization (Exact (ECS) and Inexact (ICS)); and (3) Perturbation Approach (PA). In Ref. [4] a complete and detailed analysis of various approaches for the numerical solution of vector fields is given. In this paper we extend the ideas in Ref. [4] to solve the Euler–Lagrange equations of motion for CMS.

The paper is organized as follows. In Section 2 a brief description of Euler–Lagrange equations is given. The numerical approaches for solving DAEs are discussed briefly in Section 3. The specialization of TP, ICS, ECS and PA are discussed in Section 4. Numerical comparison of these approaches for solving multibody systems are discussed in Section 5. In Section 5.1 based on these comparison and cost analysis as explained in Ref. [4] recommendations are made for a proper choice of a suitable approach.

2. Euler–Lagrange equations

We begin by formally defining the vector field that is to be solved numerically. Let $\mathcal{M}$ be an $(n-m)$ dimensional manifold in $\mathbb{R}^n$ defined by

$$g(x) = 0, \tag{2.1}$$

where $g: \mathbb{R}^n \to \mathbb{R}^m$ is a smooth function. Assume that $\exists$ an open set $\mathcal{O}$ in $\mathbb{R}^n$ containing $\mathcal{M}$ such that $g_x(x)$, the $m \times n$ Jacobian of g at x, satisfies $\operatorname{rank}(g_x(x)) = m \ \forall x \in \mathcal{O}$. Let

$$\dot{x} \triangleq \frac{dx}{dt} = f(x), \quad t \in [t_0, t_f] \tag{2.2}$$

define a vector field on $\mathcal{M}$. In other words, if $T_x\mathcal{M}$ denotes the tangent space of $\mathcal{M}$ at $x \in \mathcal{M}$, then $f(x) \in T_x\mathcal{M}$ $\forall x \in \mathcal{M}$. Usually a smooth extension of f to an open set in $\mathbb{R}^n$ containing $\mathcal{M}$ is available. We will assume this to be the case for, when dealing with numerical methods points slightly off from $\mathcal{M}$ are obtained and there may be a need to evaluate f there. Our assumption that g and f are time-invariant, i.e., they do not explicitly depend on t, is only for the purpose of simplifying the notations and some of the discussions. All the ideas of this paper easily extend to the case of time-varying g and f. Let us assume that the vector field (2.2), is solvable, i.e., given any $x_0 \in \mathcal{M}$ there exists a unique solution, $x: [t_0, t_f] \to \mathcal{M}$ that satisfies $x(t_0) = x_0$ and Eq. (2.2).

The Euler–Lagrange equation that describe the motion of CMS can be written in the form [1]

$$M(q)\ddot{q} + J^{t}(q)\lambda = Q(\dot{q}, q), \tag{2.3}$$

$$0 = \phi(q), \tag{2.4}$$

where $q \in \mathbb{R}^n$ is the vector of generalized coordinates, t is time, $M(q) \in \mathbb{R}^{n\times n}$ is the generalized mass matrix, $\phi: \mathbb{R}^n \to \mathbb{R}^m$ is a nonlinear mapping that defines the constraints (kinematical), $J = (\partial\phi/\partial q)$ is the Jacobian of ϕ with respect to q (J^{t} denotes the transpose J), $\lambda \in \mathbb{R}^m$ is the vector of Lagrange multipliers associated with the constraints, and $Q: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is a mapping that defines to generalized external forces. We make the following reasonable assumptions: (1) $m < n$; (2) M is always positive definite; (3) all the mappings in the above equation are of class C^{r+1}, $r \geqslant 2$ i.e., they are r times continuously differentiable (for most of the results in this paper this condition is stronger than necessary); (4) rank $(J) = m$ always holds (roughly, this means that all the constraints are independent).

Sometimes, the explicit dependence on $\dot{q}, q, t$ of the mappings and matrices in the above system will not be mentioned so as to simplify notations.

The system of Eqs. (2.3) and (2.4), is a DAE of index three. The index can be lowered by differentiating Eq. (2.4) twice with respect to time. This leads to the index one formulation of the Euler–Lagrange equation,

$$\begin{bmatrix} M(q) & J^{t} \\ J & \mathbf{O} \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \lambda \end{bmatrix} = \begin{bmatrix} Q(\dot{q}, q) \\ v(\dot{q}, q) \end{bmatrix}, \tag{2.5}$$

$$\phi(q) = 0, \tag{2.6}$$

$$\dot{\phi} \equiv J\dot{q} = 0, \tag{2.7}$$

where $v(\dot{q}, q) = -(\phi_q \dot{q})_q \dot{q}$.

Under the assumptions made, the linear system in $\ddot{q}$ and λ has a unique solution $\ddot{q} = f_1(\dot{q}, q), \lambda = f_2(\dot{q}, q)$, where the mappings are of class C^{r-1}. Special numerical methods for solving the linear system (2.5) are discussed in Ref. [5]. To put the equation in first order form, let $\dot{q} = v, x = (q, v)^{t}$ then we have:

$$\dot{x} = f(x), \tag{2.8}$$

$$g(x) = 0. \tag{2.9}$$

Eq. (2.8) defines a vector field on the constraint manifold defined by Eq. (2.9). The original constraints (2.4) are called position level constraints. The two new constraints introduced during the index reduction process are, respectively, called velocity level and acceleration level constraints.

3. Approaches for solving CMS equations

In this section we consider some techniques of rewriting the system (2.3) and (2.4) in an alternative form that may be easier to solve numerically. All of the different forms of the equations that we consider are equivalent in the sense that, given a consistent set of initial conditions the different forms of the system have the same analytical solution. However computationally some forms of the equations may have much different properties than others. We discuss some of the advantages and disadvantages of such a rewriting.

We can directly solve the index three form using an implicit numerical integration method such as Backward Difference Formula (BDF). The computer software ADAMS employs this technique for solving CMS. The advantage is that it is easy to formulate the system, as we do not have to differentiate the constraint or rewrite the system in any way, the sparsity of the system is preserved, and the constraints are satisfied exactly at every time step. The disadvantage is that there are several difficulties in using a variable step size BDF code for solving system in this form. Also it is difficult to obtain an accurate solution for the Lagrange multiplier λ. Further, for BDF methods, λ and $\dot{q}$ must be removed from the error estimate. A code based on this strategy does not always give reliable results for variables which are filtered out of the estimate.

A second way of solving Eqs. (2.3) and (2.4) is to differentiate the constraints twice and lower the index to one. The advantage of this technique is that we can use appropriately modified versions of some of the well known integration methods like Adams predictor–corrector and Runge–Kutta to solve the special vector field associated with CMS. But we have to pay the price of doing extra computation so that a computed solution satisfies the constraints. As mentioned earlier there are various approaches available for solving CMS. These are: (i) Parameterization approach; (ii) ICS; and (iii) ECS and (iv) PA. For complete details refer to Ref. [4].

The basic idea of PA is the following. The ODE defining the vector field on the constraint manifold is numerically integrated and after each integration step we perturb the numerical solution so as to find a point on the manifold and thus get a more accurate solution than before.

The Parameterization approach establishes a minimal set of ODEs, corresponding to the equations of motion for CMS, using a coordinate transformation (this transformation is obtained numerically) and is solved using standard ODE techniques. The solution of CMS in the original coordinate is then obtained using the inverse of the above transformation. The Coordinate partitioning approach [6] and TP approach [7] are examples of this approach.

Baumgarte [8] observed that the constraints are not satisfied in a linearly unstable fashion and it is unlikely that a subsequent numerical errors compensate this behavior. He also observed that the modified acceleration level constraint equation $\ddot{\phi} + 2\alpha\dot{\phi} + \beta^2\phi = 0$ with $\alpha > 0, \beta \neq 0$ is stable, hence implying $\dot{\phi} \approx \phi \approx 0$. This observation forms the basis of the ICS approach.

Another index two formulation of this problem can be obtained by introducing new variables μ, replacing the first equation in $\dot{q} = v$ by $\dot{q} = v + J^t\mu$ and appending the new algebraic equation $Jv = 0$ to the system (2.5)–(2.7). This new system is called the stabilized index two system. The idea of reformulating the system in this way was proposed by Gear et al. [9]. A modification of this approach, called ECS approach is discussed in Ref. [4]. The advantage of this formulation is that the position level and velocity level constraint are automatically satisfied. The extra variables are not really much of a disadvantage, because the computations can be arranged in a form that avoids the storage of the new variables and requires very little computation.

Another way is to eliminate the Lagrange multiplier λ analytically to obtain a standard ODE system (state space form or Lagrange equation of the second kind). This approach may be difficult to implement in general for very large systems, as in CMS, where the number of unknowns are usually between ten and several hundreds. In Refs. [10,11] it has been proposed to use both the original constraints and one or more of the constraints simultaneously in the DAEs formulation. This approach results in an over determined system of DAEs. The main disadvantage is the cost of linear algebra for the solution of the over determined system. This approach needs further investigation.

Let us conclude this section by making a few comments on the choice of generalized coordinates. The performance of a computer program to model these system depends strongly on the choice of the generalized coordinates q. Two fundamentally different kinds of generalized coordinates are more frequently used to derive the equations of motion. One is Cartesian, or absolute, coordinates; the other is relative, or joint, coordinates. Depending on the user need and the system at hand one has to choose the suitable coordinates.

Sophisticated approaches using relative coordinates are able to establish the state space form directly for an important class of mechanical systems called *tree configured* systems. This form can be numerically treated by standard ODE methods, and is of lower dimension than the DAE form. However, the corresponding matrices are normally dense [6]. For systems which are not tree configured, there are loop closure conditions which remain as algebraic

equations and cannot in general be solved analytically. For these systems usually the state space from is established numerically [6]. This is easy to do in the case of linear constraints, but in the general case the coordinates may change from integration step to integration step or even in between. The choice of coordinates directly influence both the number of equations and their order of nonlinearity. Furthermore, depending upon the form of the equations one method of numerical integration may be preferable to another in terms of efficiency and accuracy. A comparison between the choice of coordinates is given in Ref. [12], Table 1.1, p. 12.

4. Specialization of various approaches to CMS

In the following sections we extend the ideas presented in [4] for solving vector fields to simulation of CMS. In this paper we consider the following approaches (i) TP approach; (ii) ICS, ECS approach and (iii) PA.

4.1. Specialization of TP for CMS

Because of its good performance and ease of implementation, we only specialize the TP approach to CMS. Ideas associated with the specialization of Coordinate Partitioning approach to CMS, and also its performance are similar to those of TP. The idea of TP approach was introduced to solve CMS equations by Mani et al. [7].

In this approach we need to choose two orthogonal matrices U_0 and V_0 to compute local parameterization [4]. Here we give a brief description of Procedure TP. For complete details, refer to Ref. [4].

Procedure TP. Computing $\tilde{f}(y)$ given y

Step 1: Solve $\tilde{g}(y,z) = 0$ for z

Step 2: Set $x = \psi(y) = x_0 + U_0 y + V_0 z$; (diffeomorphism)

Step 3: Compute $f(x)$ and set $\tilde{f}(y) = U_0^{\mathrm{t}} f(x)$.

Here we will discuss an efficient choice of U_0 and V_0 using the QR decomposition of J^{t}. Let the QR decomposition of $J^{\mathrm{t}} = Q(R_1, O)^{\mathrm{t}}$, where $R_1 \in \mathbb{R}^{m\times m}$. Let Q be partitioned as $[V:U]$, $V \in \mathbb{R}^{n\times m}$, $U \in \mathbb{R}^{n\times(n-m)}$. From the above QR decomposition, we have $J^{\mathrm{t}} = VR_1$. This shows that the columns of V form an orthonormal basis of the column space of J^{t} or the row space of J. Premultiplying by U^{t} we get $JU = 0$. Thus, the columns of U are base vectors for the null space of J. The null space and row space of J are orthogonal subspaces of the configuration space $\mathbb{R}^n$. Since the row space of J is normal to the associated constraint surface, the null space of J spans the tangent plane of the the constraint surface. So the columns of U form an orthogonal basis of the constraint tangent plane.

$JJ^t = R_1^t R_1$ is the Cholesky factorization of JJ^t and is positive definite ($\text{rank}(J) = m$). Thus R_1 is unique and we have $V = J^t R_1^{-1}$. This shows that V is also unique. However, U is not an unique matrix since a linear combination of the columns of U are also an orthonormal basis normal to V. The QR factorization can be computed using any of the following techniques: (i) Householder transformation; (ii) Givens transformation; and (iii) Gram–Schmidt orthogonalization process. Mani et al. [7] have used singular value decomposition method which uses Householder transformation to get the above orthogonal matrices U and V.

Let us now briefly explain this approach as applied to CMS. Let $Y = (y, \dot{y})^t$ and $Z = (z, \dot{z})^t$; $y \in \mathbb{R}^{n-m}$, $z \in \mathbb{R}^m$, $y, \dot{y}$ are namely the independent position and velocity variables (vectors in the tangent plane) and $z, \dot{z}$ are namely the dependent position and velocity variables (vectors in the normal plane). In terms of y and z, $q = q_0 + Uy + Vz$. Differentiating q with respect to time and combining the equations, we get equation of the form

$$x = \tilde{x}_0 + \mathscr{U} Y + \mathscr{V} Z \equiv \Psi(Y). \tag{4.1}$$

The diffeomorphism Ψ is given by Eq. (4.1) and $\mathscr{U} = \text{diag}(U, U)$ and $\mathscr{V} = \text{diag}(V, V)$ play the roles of U_0 and V_0, respectively. We compute $f(x)$ and set the ODE for Y as

$$\dot{Y} = \mathscr{U}^t f(x) \triangleq \tilde{f}(Y). \tag{4.2}$$

Once Eq. (4.2) is integrated using the initial value as $Y_0 = Y(t_0) = (0, U^t \dot{q}_0)^t$, we can compute $x(t) = \Psi(Y(t))$ using Eq. (4.1). The following procedure explains the steps involved in TP.

Procedure TP-CMS. Computing $\tilde{f}(Y)$ given Y

Step 1: Set $Y = (y, \dot{y})^t$ and $Z = (z, \dot{z})^t$ such that $q = q_0 + Uy + Vz$ and $v = U\dot{y} + V\dot{z}$. Compute Z as follows. Using q we have $\phi(q_0 + Uy + Vz) = 0$. Re-framing this as a function (ρ) of the unknown z we have $\rho(z) = 0$. We solve for z from using the MNR iteration procedure. Using the velocity level constraint $Jv = 0$ can be written as $A\dot{z} = c$, where $A = JV$, $c = -[JU\dot{y}]$. This linear system is solved for $\dot{z}$ by doing LU decomposition of A ($\text{rank}(J) = M$) and set $Z = (z, \dot{z})^t$.

Step 2: Evaluate $\Psi(Y(t)) = x(t)$, i.e., $q = q_0 + Uy + Vz$ and $v = U\dot{y} + V\dot{z}$.

Step 3: Compute $f(x)$ and set $\tilde{f}(Y) = \mathscr{U}^t f(x)$.

For tangential parameterization, Mani et al. [7] use an interesting idea to decide when a change in parameterization is needed. To determine when to compute a new parameterization, i.e., to redefine the orthogonal matrix Q, we have used the following criteria in our implementation. Premultiplying $v = U\dot{y} + V\dot{z}$ with v^t we get $v^t v = \dot{y}^t \dot{y} + \dot{z}^t \dot{z}$. Under the assumption $\phi_t = 0$, Mani et al. [7] proposed a criterion to decide when a change in parameterization is needed.

Roughly the idea is as follows. At t_0 it is clear that the l_2 norm of the velocity vector $\dot{y}$ equals the l_2 norm of the velocity vector v, since the l_2 norm of the $\dot{z}$ is zero. At other points on the constraint manifold this will be violated. Define the proportion between the l_2 norms as $\|\dot{y}\|_2/\|v\|_2$. This ratio gives a good indication of a need for a change in parameterization. Choosing an appropriate $\mu \in (0,1)$, say, $\mu = 0.5$, a change in parameterization is called for when the ratio becomes less than μ.

It is important to note that we need to specify the integration tolerances while integrating Eq. (4.2) in terms of Y. It is natural to specify the integration tolerances in terms of q and v, since we are originally solving the Euler–Lagrange equation. So we need to transform this tolerances in terms of the integrated variable Y. As explained in Ref. [4] we compute $\Psi_Y = \text{diag}(U - V(JV)^{-1}JU)$, so that $\|e_x\|_W \approx \|\Psi_Y e_Y\|_W \leqslant \tau$. This has to be done very efficiently since it has to be computed at every integration step. Also note that it involves A^{-1}. After the completion of Step 3 of the algorithm we have the LU decomposition of A. Writing $B = A^{-1}J$ we solve for B as follows. Solve $(LU)b_i = j_i$, $1 \leqslant i \leqslant n$ for b_i, where b_i and j_i are the columns of B and J, respectively. It can be easily solved by forward and backward substitutions.

4.2. Specialization of CS for CMS

In this section we briefly discuss the CS approach originally suggested by Baumgarte [8] to solve the equations of CMS. This approach will be referred as ICS. The *stabilized index two* problem [9] will be discussed with some modifications and the implementation details are also explained. We will call this approach as ECS [4].

4.2.1. Inexact constraint stabilization

The second time derivative of the constraint equation $\phi(q) = 0$ can be written as $\ddot{\phi} \equiv J\ddot{q} - v = 0$. From control theory, it is well known that even with small perturbations from zero, the solution of the equation $\ddot{\phi} = 0$ can become unstable; that is it can lead to values of ϕ and $\dot{\phi}$ that are far from zero. Baumgarte [8] observed that the modified acceleration equation

$$\ddot{\phi} + 2\alpha\dot{\phi} + \beta^2\phi = 0 \tag{4.3}$$

with $\alpha > 0$ and $\beta \neq 0$ is stable, hence implying $\dot{\phi} \approx \phi \approx 0$ even with perturbations of ϕ. This observation forms the basis of the ICS presented by Baumgarte. Eq. (4.3) becomes $J\ddot{q} = v - 2\alpha\dot{\phi} - \beta^2\phi = \hat{v}$.

This equation along with the dynamical equation will form the linear system

$$\begin{bmatrix} M(q) & J^t \\ J & \mathbf{O} \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \lambda \end{bmatrix} = \begin{bmatrix} Q(q,\dot{q}) \\ \hat{v}(q,\dot{q}) \end{bmatrix}$$

and is solved. This leads to a second order ODE (which is equivalent the CMS vector field on $\mathcal{M}$), which is then solved numerically using ODE techniques. The advantages of this approach are that it is simple and can easily be coded. However, it suffers from important defects such as: (i) a lack of systematic way to choose α and β; (ii) the effect of α and β on the accuracy of the numerical solution; and (iii) the effect of α and β on stiffness. The notion of stiffness in the context of DAE is explained in Ref. [13].

4.2.2. Exact constraint stabilization

We are concerned with the numerical solution of the system:

$$\begin{aligned} &\dot{q} = v, \\ &M(q)\dot{v} = Q(q, v) - J^{t}\lambda, \\ &\phi(q) = 0. \end{aligned} \tag{4.4}$$

The assumption that ϕ is time invariant is only for notational convenience and all the discussion can be easily extended to the time varying case. Gear et al. [9] suggested an index two formulation of the system (4.4) by introducing new variables μ, replacing the first equation in Eq. (4.4) by $\dot{q} = v + J^{t}\mu$ and appending the new algebraic equation $Jv = 0$ to the system i.e., we get:

$$\begin{aligned} &\dot{q} = v + J^{t}\mu, \\ &M(q)\dot{v} = Q - J^{t}\lambda, \\ &\phi(q) = 0, \\ &Jv = 0 \end{aligned}$$

and this system was called stabilized index two problem. (Note that the index of Eq. (4.4) is three.) Also it was proved that if J is full rank, then any solution of the original index three system (4.4) is a solution of the stabilized index two system, and conversely. The advantage of this formulation is that the velocity level constraints are also enforced, thus eliminating the problem of drift of these constraints. The extra variables are not really much of a disadvantage, because the computation can be arranged in an efficient way. This way of solving which avoids unnecessary computations and storage locations will be explained now.

The stabilized index two system can be rewritten as:

$$\dot{q} = v + \bar{\mu}, \tag{4.5}$$

$$\dot{v} = M^{-1}Q(q, v) - M^{-1}J^{t}\lambda, \tag{4.6}$$

$$\phi(q) = 0, \tag{4.7}$$

$$Jv = 0 \tag{4.8}$$

where $\bar{\mu} = M^{-1}J^{t}\mu$. The reason for the choice of $\bar{\mu}$ will be explained later. While implementing this approach the special structure of Eqs. (4.5)–(4.8) should be

exploited fully. The following procedure describes the basic steps involved in an integration step from time t_n to t_{n+1}.

Procedure ECS. Stabilized index two formulation

Step 1: Predict q and v at time t_{n+1} from the previous values and obtain q^p_{n+1} and v^p_{n+1}.

Step 2: Evaluate $Q(q^p_{n+1}, v^p_{n+1})$ and $J(q^p_{n+1})$.

Step 3: Solve Eqs. (4.5) and (4.7) for $\bar{\mu}_{n+1}$ and q_{n+1} by MNR iteration, keeping $v_{n+1} = v^p_{n+1}$ and using an appropriate BDF for $\dot{q}$ and $\dot{v}$.

Step 4: Solve Eqs. (4.6) and (4.8) for λ_{n+1} and v_{n+1} using the same BDF. Here λ_{n+1} is solved directly from a linear set of equations.

Step 5: Repeat steps 2–4 once, using the latest computed values of q_{n+1} and v_{n+1} for q^p_{n+1} and v^p_{n+1}. This amounts to a single corrector iteration.

It can be proved [10] that the numerical process consisting of steps 1 and 2, one iteration of steps 3 and 4 is convergent. We note that this algorithm involves the solution of an $(m \times m)$ linear system is step 3 for each MNR iteration, and a further linear system in step 4.

It is very important to note that, in the above procedure, we have decoupled the nonlinear system at position and velocity levels. The solutions of the nonlinear subsystems are then corrected through functional iterations. We might also consider solving the nonlinear system by MNR method, but treat the linear system which must be solved at each step by special techniques which exploit the available sparsity. The idea is to decouple at the level of linear equations at each MNR iteration. It is shown in Ref. [10] the decoupling at the level of the nonlinear equations or at the level of linear equations at each MNR iteration results in similar convergence conditions. We will now discuss the implementation details of ECS. Before going into details, let us first briefly discuss about BDF method mentioned in the algorithm.

There are three main approaches to extend the fixed step size multi-step methods to variable step size. They are: (1) fixed coefficient; (2) variable coefficient; and (3) fixed leading coefficient. The relative advantages and disadvantages of the various BDF methods are explained in great detail in Ref. [14]. As suggested in Ref. [14] we employ the variable step size variable order fixed leading coefficient implementation of BDF method to advance the solution from one time step to the next. That is, the method approximates the derivative using the kth order BDF, where $0 \leqslant k \leqslant 5$. On every step, it chooses the order k, and step size, h_{n+1}, based on the behavior of the solution. The well-known DAE code DASSL [15] employs this method.

We will now describe the basic formulae used to solve an ODE, say, $\dot{y} = f(y)$. Suppose we have approximations y_{n-i} to the solution $y(t_{n-i})$ for $i = 0, 1, \ldots, k$, where k is the order of the BDF method used. We would like to find an approximation to the solution at time t_{n+1}. First, an initial guess

for the solution and its derivative at t_{n+1} is formed by evaluating the predictor polynomial and the derivative of the predictor polynomial at t_{n+1}. Shampine [16] explains clearly the reasons for evaluating the derivative as the derivative of the predictor polynomial. The approximation y_{n+1} to the solution at t_{n+1} which is finally accepted is the solution of the corrector formula. The formula used is the fixed leading form of the kth order BDF method.

The values of the predictor $y^p_{n+1}, \dot{y}^p_{n+1}$ and the corrector y_{n+1} at t_{n+1} are defined in terms of polynomials which interpolate the solution at previous time points. Following the ideas of Shampine and Gordon [17] these polynomials are represented in terms of modified divided difference. The corrector formula can be written as [14]

$$\dot{y}_{n+1} = \dot{y}^p_{n+1} - \left(\frac{\alpha_s}{h_{n+1}}\right)(y_{n+1} - y^p_{n+1}), \tag{4.9}$$

where the fixed leading coefficient α_s is defined as $\alpha_s = -\sum_{j=1}^{k}(1/j)$. We will now explain the five steps of the algorithm described earlier.

Detailed Procedure ECS

Step 1: Predict q and v at t_{n+1} as q^p_{n+1} and v^p_{n+1}.

Step 2: Evaluate $Q(q^p_{n+1}, v^p_{n+1})$ and $J(q^p_{n+1})$.

Step 3: Solve Eqs. (4.5) and (4.7) for $\bar{\mu}_{n+1}$ and q_{n+1} by MNR iteration, keeping $v_{n+1} = v^p_{n+1}$ and using the BDF method mentioned above. From Eqs. (4.5) and (4.9), we have $q_{n+1} = q^p_{n+1} + \Delta q$, where

$$\Delta q = \left(\frac{h_{n+1}}{\alpha_s}\right)\left(\dot{q}^p_{n+1} - [v_{n+1} + (M^{-1}J^t)_{n+1}\mu_{n+1}]\right)$$

and $(M^{-1}J^t)_{n+1} = M^{-1}(q_{n+1})J^t(q_{n+1})$. We solve for μ_{n+1} as follows. Let $\gamma = (\mu_{n+1}h_{n+1}/\alpha_s)$ and Eq. (4.7) can be written as $\phi(\gamma) = 0$. We solve for γ from this relation using MNR iteration process given by

$$\gamma^{l+1} - \gamma^l = -\left(\frac{\partial\phi}{\partial\gamma}\right)^{-1}\phi(\gamma), \tag{4.10}$$

where $(\partial\phi/\partial\gamma) = J(\partial q_{n+1})/\partial\gamma$ and $(\partial q_{n+1}/\partial\gamma) = -(I + (\partial(M^{-1}J^t)/\partial q_{n+1})\gamma)^{-1}\, M^{-1}J^t$. Here we make an assumption that the rate of change of $M^{-1}J^t$ with respect to q_{n+1} is negligible. This will save the over-head cost of each integration step and this assumption may at the most lead to slow convergence. Using the above assumption in Eq. (4.10) can be written as

$$H\delta\gamma = \phi, \tag{4.11}$$

where $H = JM^{-1}J^t$, $\delta\gamma = \gamma^{l+1} - \gamma^l$. Since $\text{rank}(J) = m$ and M is positive definite, $\text{rank}(H) = m$. We solve Eq. (4.11) for $\delta\gamma$ and γ is updated as $\gamma := \gamma + \delta\gamma$. Update q_{n+1}. The iteration is started with $\gamma^0 = 0$.

Step 4: Evaluate $Q(q_{n+1}, v_{n+1})$. Solve for λ_{n+1} and v_{n+1} form Eqs. (4.6) and (4.8) using the same BDF method. From Eqs. (4.6) and (4.9), we have $v_{n+1} = v^p_{n+1} + \Delta v$, where

$$\Delta v = \left(\frac{h_{n+1}}{\alpha_s}\right)\left(\dot{v}^p_{n+1} - M^{-1}Q_{n+1} + M^{-1}J^t\lambda_{n+1}\right)$$

and $Q_{n+1} = Q(q_{n+1}, v_{n+1})$. Substituting v_{n+1} in Eq. (4.8) we get

$$Hc = b, \tag{4.12}$$

where $c = (\lambda_{n+1}h_{n+1}/\alpha_s)$; $b = -Jd$; $d = (h_{n+1}/\alpha_s)(\dot{v}^p_{n+1} - M^{-1}Q_{n+1}) + v^p_{n+1}$. The linear system (4.12) is solved for c. While solving Eqs. (4.11) and (4.12), the same factorization of H should be used. This explains the choice of $\bar{\mu}$ as $M^{-1}J^t\mu$. Also from the above equations we have $v_{n+1} = d + M^{-1}J^tc$. The reason for doing the computation in this fashion is to reduce the over head cost of the integration step.

Step 5: Repeat steps 2–4 using the latest values of q_{n+1} and v_{n+1}. This amounts to a single corrector iteration.

Remark 4.1. Note that $H = JM^{-1}J^t$. Computing H this way is not efficient since it involves three matrix multiplications. We write $H = AA^t$ where $A = JR$ and R is Cholesky factor of M (M is positive definite and $M = R^tR$). From $R^tA^t = J^t$ we solve for A^t as follows. Solve $R^ta_i = j_i$, $1 \leqslant i \leqslant m$, where a_i and j_i are the columns of A^t and J^t, respectively. For each i this system is a triangular system of equations and can be solved easily by forward and backward substitutions. We do a skinny QR decomposition of A^t as $A^t = Q_1R_1$, so that $H = R_1^tR_1$ and the two $(m \times m)$ linear systems (4.11) and (4.12) are divided into two triangular subsystems which can be solved easily by forward and backward substitutions.

4.3. Specialization of PA to CMS

In this section we extend the ideas of the PA to the solution of the special vector field associated with CMS.

Let us briefly describe PA as applied to general vector fields [4]. In the PA, a correction is applied to a numerical solution of Eq. (2.2) after each integration step so as to satisfy Eq. (2.1). To describe the approach, it is sufficient to say what is done in one integration step. Suppose k steps of the numerical solution of Eqs. (2.1) and (2.2) have been done and $t = t_k$ has been reached. Let $x_k \in \mathcal{M}$ be the solution approximant at $t = t_k$. Denote the local solution by $x(\cdot)$, i.e., $x(\cdot)$ is the solution of Eq. (2.2) with $x(t_k) = x_k$.. Let τ denote the integration tolerance. In the $(k+1)$st step, the aim is to determine a step size h_k and an $x_{k+1} \in \mathcal{M}$ that satisfy

$$\|x(t_{k+1}) - x_{k+1}\| \leqslant \tau, \tag{4.13}$$

where $t_{k+1} = t_k + h_k$. The determination of h_k and x_{k+1} is described by the following procedure.

Procedure PA Determination of h_k and x_{k+1} by the PA

Step 1: Numerically integrate Eq. (2.2) from $x(t_k) = x_k$ using local error control (without taking into account Eq. (2.1)) to obtain a step size h_k and an approximant, $\bar{x}_{k+1}$ that satisfy

$$\|x(t_{k+1}) - \bar{x}_{k+1}\| \leqslant \tau/2. \tag{4.14}$$

Step 2: Solve the optimization problem

$$\min \|x - \bar{x}_{k+1}\|, \quad \text{s.t. } g(x) = 0, \tag{4.15}$$

and set x_{k+1} = the minimizer of Eq. (4.15).

Remark 4.2. Step 2 of Procedure PA is stronger than what is really needed. It is sufficient if x_{k+1} satisfies

$$g(x_{k+1}) = 0, \quad \|x_{k+1} - \bar{x}_{k+1}\| \leqslant \tfrac{1}{2}\tau. \tag{4.16}$$

From $\|x_{k+1} - x(t_{k+1})\| \leqslant \|x_{k+1} - \bar{x}_{k+1}\| + \|x(t_{k+1}) - \bar{x}_{k+1}\| \leqslant 2\|x(t_{k+1}) - \bar{x}_{k+1}\| \leqslant \tau$ and Eq. (4.14), it follows that x_{k+1} also satisfies Eq. (4.13). Though the feasibility problem (4.16) is usually as difficult to solve as the optimization problem (4.15), it is useful because its solution is simpler to verify.

Let us extend the ideas of the PA to the solution of the special vector field associated with CMS. There are two key computations which benefit from the special structure: (i) the evaluation of $f(x)$; and (ii) the manifold-correction in step 2 of procedure PA. The efficient evaluation of $f(x)$ is explained in [5]. Let us now elaborate on item (ii). Usually, scaled l_2 and l_∞ norms are the popular norms used for measuring integration errors. Shampine suggests a method for solving Eq. (4.15) when the scaled l_2 norm is used. Further, he makes the remark that, "if the ODE solver is based on the maximum norm l_∞, one might prefer to alter it to use the Euclidean norm l_2 so as to arrive at a linear algebra problem which has a classical solution". For the scaled l_2 norm we suggest a method which is slightly different but more efficient than Shampine's method. A method for the scaled l_∞ norm which is cheaper than that of the l_2 norm solution is discussed in Ref. [1].

Consider the scaled l_2 norm first. Here $\|x\| = \sqrt{x^t W x}$, where W is a symmetric positive definite matrix. Typically, W is a diagonal matrix, the diagonal elements representing the variable weights applied to the different components of x. To simplify the notations, let us assume the objective function in Eq. (4.15) is replaced by half its square, and an appropriate coordinate transformation,

$$x = \bar{x}_{k+1} + TX \tag{4.17}$$

(T is a nonsingular matrix such that $T^t W T = I_n$) is done, so that Eq. (4.15) becomes

$$\min \|X\|_2^2/2 \quad \text{s.t. } G(X) = 0, \tag{4.18}$$

where $G(X) = g(\bar{x}_{k+1} + TX)$.

The first order necessary conditions for Eq. (4.18) are $X - G_X^t(X)\mu = 0$, $G(X) = 0$, where μ is the Lagrangian multiplier for equality constraint. We can use the MNR method to obtain a solution X^*.

We will describe the specialization only for the scaled l_2 norm. The ideas can be easily extended to the scaled l_∞ norm. Let $\dot{q} = v, x = (q, v)^t$ and $\|x\|$, the integration norm on x, be defined by $\|x\|^2 = q^t W_1 q + v^t W_2 v$, where W_1 and W_2 are positive definite matrices. Compute $n \times n$ nonsingular matrices T_1 and T_2 such that $T^t W_1 T_1 = T_2^t W_2 T_2 = I_n$. (Usually W_1 and W_2 are diagonal weighting matrices and so the computation of T_1 and T_2 is easy.) Let $T =$ block diag$\{T_1, T_2\}$, $\bar{x}_{k+1} = (\bar{q}, \bar{v})^t$ and $X = (Q, V)^t$ so that Eq. (4.17) becomes $q = \bar{q} + T_1 Q$, $v = \bar{v} + T_2 V$. Also, Eq. (4.18) becomes,

$$\min (\|Q\|_2^2 + \|V\|_2^2)/2 \quad \text{s.t. } G_1(Q) = 0, \ G_2(Q, V) = 0, \tag{4.19}$$

where $G_1(Q) = \phi(\bar{q} + T_1 Q), G_2(Q, V) = J(\bar{q} + T_1 Q)[\bar{v} + T_2 V]$.

As mentioned in Remark 4.2 an exact solution of Eq. (4.19) is not necessary. This allows a simplifying assumption to be made on Eq. (4.19). Typically $J(q)$ is a slowly varying function of q. Also, since the value of Q in the solution of Eq. (4.19) is small ($\|Q\|_2 \leqslant \tau/2$ so that q varies over a small range) we can replace $G_2(Q, V)$ in Eq. (4.19) by $\tilde{G}_2(V) = J(\bar{q})[\bar{v} + T_2 V]$.

The explicit dependence on time t for G_1 and G_2 to simplify the notations. It is then easy to see that Eq. (4.19) gets decomposed into the following smaller problems:

$$\min \|Q\|_2^2/2, \quad \text{s.t. } G_1(Q) = 0, \tag{4.20}$$

$$\min \|V\|_2^2/2, \quad \text{s.t. } \tilde{G}_2(V) = 0. \tag{4.21}$$

Each of these problems can be solved using MNR method. The MNR iterations require the skinny QR decomposition of $(G_1)_Q(0) = J(\bar{q})T_1$ and $(\tilde{G}_2)_V(0) = J(\bar{q})T_2$. Since $\tilde{G}_2$ is an affine function of V one MNR iteration will yield the exact solution of Eq. (4.21). The solution of Eq. (4.20) may require more than one iteration. A particular assumption on the integration norm helps to improve the efficiency significantly.

Assumption 4.3. W_1 and W_2, the weighting matrices for the position and velocity vectors, satisfy the relation $W_2 = \alpha^2 W_1$, where α is a nonzero scalar having the units of time.

This is a reasonable assumption because it simply requires that the relative weighting among the position variables is same as the the relative weighting among the velocity vectors. Then $T_2 = (1/\alpha)T_1$. Therefore, if $J(\bar{q})T_1 = Q_1 R_1$ is skinny QR decomposition of $J(\bar{q})T_1$ then $J(\bar{q})T_2 = Q_2 R_2$ with $Q_2 = Q_1$ and

$R_2 = (1/\alpha)R_1$ is the skinny QR decomposition of $J(\bar{q})T_2$, and so a separate computation of the decomposition of $J(\bar{q})T_2$ is unnecessary.

Suppose W_1 and W_2 are user specified diagonal weighting matrices, but they do not satisfy $W_2 = \alpha^2 W_1$. The following scheme gives a good way of choosing $\widetilde{W}_2$ such that: $\widetilde{W}_2 = \alpha^2 W_1$ for some α; and if W_2 is replaced by $\widetilde{W}_2$ then the user defined error tolerances will be met.

We will combine the above ideas into a Manifold Correction Algorithm denoted by C^* as follows.

Manifold correction algorithm C^*

(a) Start from the point $\bar{x}$, provided by the Step 1 of the Procedure PA.
Set $l = 0, X^l = [Q, V]^t = 0, \mu_1 = 0$

(b) Compute $J(\bar{q}), T_1$, and T_2.

Start the MNR iteration process.

(a) IF $l = 0$ THEN
- Evaluate $A = JT_1$
- Do the QR factorization of A^t, i.e., $A^t = Q_1 R_1$

ENDIF

(b) Compute $RHS1 = G_1(Q^l) = \phi(\bar{q} + T_1 Q^l)$

(c) Solve: $R_1^t R_1 \delta\mu_1 = RHS1$ for $\delta\mu_1$ by solving a pair of triangular systems.

(d) Set:
- $\mu_1^{l+1} := \mu_1^l + \delta\mu_1$
- $Q^{l+1} = A^t \mu_1^{l+1}$
- $q := \tilde{q} + T_1 Q^{l+1}$
- $l := l + 1$

(e) Check the corrector failure flag C. (Failure rarely occurs, since $\bar{x}$ provided by step 1 of the Procedure PA is very good.)
If(C = F) then indicate Step Failure and go back to Procedure PA with a reduced step size;
Else CONTINUE
Compute $RHS2 = \alpha^2 \widetilde{G}_2(V) = \alpha^2 J(\bar{q})[\bar{v} + T_2 V]$

Solve: $R_1^t R_1 \mu_2 = RHS2$ for μ_2 by solving a pair of triangular systems.

(a) $V := (1/\alpha) A^t \mu_2$

(b) $v := \tilde{v} + T_2 V$

5. Numerical examples

Clearly, it is not possible to single out a particular approach as the most suitable for solving any vector field. The choice of an appropriate approach-integration method combination for a given application has to be made by looking at special structures in the vector field, by doing the cost analysis [4] and, very importantly, by studying the performance of various combinations on selected

numerical examples. Here we only consider vector fields associated with CMS. The four approaches discussed in this paper are implemented in FORTRAN. A computer code called Dynamic Analysis of MEchanical Systems (DAMES) was developed for this purpose.

A carefully considered and explicitly stated experimental design is crucial in making valid inferences about the performance of the mathematical software. Developing a sound experimental design involves identifying the variables expected to be influential in determining the code performance, deciding the appropriate measures of performance. Choosing the appropriate performance indicators is a crucial factor in computational experiments. We have chosen performance indicators which are as independent as possible of the problem at hand. The following performance indicators are common to all the approaches: CPU–CPU Time (calculated in Micro Vax – under ULTRIX-32m version 1.2 OS); NF-Number of Function evaluations (f-evaluations); NS-Number of integration steps to complete the integration from t_0 to t_f, NJ-Number of J matrix evaluation; NI-Average number of iterations taken to solve the nonlinear square system; and Nϕ-Number of ϕ evaluations. Apart from these, the indicators which are specific to particular approach are: NFK-Number of LU factorizations (step 1 of Procedure TP); NT-Number of triangular systems solved (step 1 of Procedure TP, solving for μ in the PA, step 3 of Procedure ECS); NP-Number of new TP done. Before discussing the examples let us make the following remark.

Remark 5.1. The selection of the stabilizing parameter α in the case of ICS approach is a crucial issue and there is no clear general way of choosing it is available. So in all the examples the parameter α is chosen in the following way. We start with $\alpha = 0$ (i.e., integrating just the underlying ODE) and slowly increase its value. By looking at the solution plots a proper (optimal) α is chosen. So while actually comparing the performance indicators of this approach with the other approaches it is very important to keep in mind about the huge exercise involved in choosing this optimal α.

In the case of TP approach, choosing the ratio μ is very important. Number of new parameterizations done depends mainly on this optimal choice of μ. The number of new parameterizations done also depends on ρ, the rate of convergence [4].

Example 5.2 (*Slider-crank mechanism*). Consider the elementary model of a slider–crank mechanism shown in Fig. 1, where the units of length are in meters. The crank (body-1) rotates without friction about an axis perpendicular to the x–y plane. The connecting rod (body-2) is constrained so that its center point slides without friction along the x axis. The polar moment of inertia of bodies 1 and 2 are J_1 and J_2 kg-m^2, respectively, and the mass of body-2 is m_2 (kg). A constant torque $T = 10$ Nm is applied to body-1. The

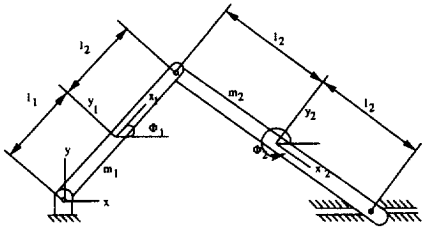

Fig. 1. Elementary slider-crank mechanism.

generalized coordinate vector $q = (\phi_1, x_2, \phi_2)^t$. Let us now define the inputs to DAMES.

$$M(q) = \text{diag}(J_1, m_2, J_2), \qquad J(q) = \begin{bmatrix} -\sin\phi_1 & -1 & -2\sin\phi_2 \\ \cos\phi_1 & 0 & 2\cos\phi_2 \end{bmatrix},$$

$$\phi(q) = \begin{bmatrix} \cos\phi_1 + 2\cos\phi_2 - x_2 \\ \sin\phi_1 + 2\sin\phi_2 \end{bmatrix}, \qquad Q(\dot{q}, q) = (T, 0, 0)^t,$$

$$v(\dot{q}, q) = \begin{bmatrix} \cos\phi_1\,\dot{\phi}_1^2 + 2\cos\phi_2\,\dot{\phi}_2^2 \\ \sin\phi_1\,\dot{\phi}_1^2 + 2\sin\phi_2\,\dot{\phi}_2^2 \end{bmatrix}$$

Time interval $[t_0, t_f] = [0, 2]$ s, $J_1 = 1$, $J_2 = 2$, $m_2 = 2$. The CIC are $\phi_1 = \pi/4$ rad, $x_2 = 2.5779$ m, $\phi_2 = -0.3613$ rad and zero Initial Velocity. This system is simulated using the four approaches PA, TP, ECS, ICS and the results are tabulated in Table 1.

Example 5.3 (*Three link cylindrical coordinate manipulator*). In this a three link coordinate cylindrical manipulator shown in Fig. 2 is considered. We assume that the joints of the robot are all rigid.

The end of the robot is constrained to move in one-dimensional path, namely a circle in the x–z plane with y being a constant. The constraint function in Cartesian coordinates is given by $(y - 0.169 = 0,\ x^2 + z^2 - d = 0)^t$, $d =$

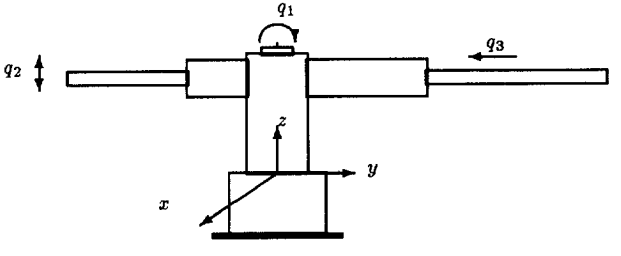

Fig. 2. Three link cylindrical coordinate manipulator.

Table 1
Slider–crank mechanism

	PA	TP	ECS	ICS
TOL = 10^{-6}				
NF	286	200	–	266
NS	145	100	580	135
NJ	291	455	468	271
Nϕ	233	468	3372	271
CPU	5.7	6.3	7.9	3.8
NI	1.61	2.1	2.2	–
NT	748	693	4817	–
NFK	–	225	–	–
NP	–	4	–	–
TOL = 10^{-4}				
NF	162	121	–	166
NS	83	61	398	83
NJ	167	275	386	171
Nϕ	126	287	2340	171
CPU	3.0	3.9	5.1	1.5
NI	1.51	1.97	2.2	–
NT	410	423	3100	–
NFK	–	136	–	–
NP	–	2	–	–
TOL = 10^{-2}				
NF	68	55	–	68
NS	36	28	186	36
NJ	73	143	172	73
Nϕ	63	165	988	73
CPU	1.8	1.9	3.1	1.0
NI	1.75	1.97	2.3	–
NT	190	234	1686	–
NFK	–	70	–	–
NP	–	2	–	–

0.221423. The generalized coordinate vector $q = (q_1, q_2, q_3)^t$ as shown in Fig. 2. The inputs are: $M(q) = \text{diag}(J_1 + J_2 + J_3 + m_3(q_3 + l_3)^2,\ m_2 + m_3,\ m_3)$

$$Q(\dot{q}, q) = \begin{bmatrix} 2m_3(q_3 + l_3)\dot{q}_1\dot{q}_3 \\ (m_2 + m_3)g \\ -m_3(q_3 + l_3)\dot{q}_1^2 \end{bmatrix},$$

$$J(q) = \begin{bmatrix} (q_3 + l_3)\cos q_1 & 0 & \sin q_1 \\ -2(q_3 + l_3)^2 \cos q_1 \sin q_1 & 2q_2 & 2(q_3 + l_3)\cos^2 q_1 \end{bmatrix},$$

$$\phi(q) = \begin{bmatrix} (q_3 + l_3)\sin q_1 - 0.169 \\ (q_3 + l_3)^2 \cos^2 q_1 + q_2^2 - d \end{bmatrix},$$

$$v(\dot{q},q)=\begin{bmatrix}(q_3+l_3)\sin q_1\,\dot{q}_1^2-2\cos q_1\,\dot{q}_1\dot{q}_3\\ 2(q_3+l_3)^2\cos 2q_1\,\dot{q}_1^2-2\dot{q}_2^2-2\cos^2 q_1\,\dot{q}_3^2+\\ 4(q_3+l_3)\sin 2q_1\,\dot{q}_1\dot{q}_3\end{bmatrix}.$$

Time interval $[t_0,t_f]=[0,4]$ s, $m_2=1$ kg,$m_3=2$ kg, $J_1=0.1$ kg-m^2, $J_2=0.2$ kg-m^2, $J_3=0.1$ kg-m^2, $l_3=0.2$ m CIC $q_1=0.48607$, $q_2=0.34512$, $q_3=0.16176$ and zero initial velocity. The control law for stabilizing the system to the equilibrium point, $q_e=(0.436,\ 0.3,\ 0.2)^t$, designed using a linearized approach [18] is given by

$$\begin{aligned}u=&\,0.363-3.655(q_1-0.436)-4.181(q_2-0.3)+3.136(q_3-0.2)\\&-0.455\dot{q}_1-2.167\dot{q}_2+1.083\dot{q}_3\\&29.4-15.938(q_1-0.436)-18.229(q_2-0.3)+13.672(q_3-0.2)\\&-2.167\dot{q}_1-10.325\dot{q}_2+5.163\dot{q}_3\\&0.423+8.267(q_1-0.436)+9.455(q_2-0.3)-7.091(q_3-0.2)\\&+1.083\dot{q}_1+5.163\dot{q}_2-2.581\dot{q}_3.\end{aligned}$$

This system is simulated using the four approaches PA, TP, ECS, ICS and the results are tabulated in Table 2.

Example 5.4 (*Quick-return mechanism*). As an example of the many compound mechanisms that arise in practice, the quick-return mechanism of Fig. 3 that represents a shaper is considered. With counter-clockwise rotation of the crank (body-3), cutting occurs as the tool (body-6) moves to the left through the workpiece. The quick-return stroke of the tool occurs as it moves to the right. In the model of Fig. 3, each link is modeled as a body. The elements of the model are as follows: Body-1 is ground, and the body-fixed frames are as shown in the figure and the constraints are defined in Table 3. The generalized coordinate vector $q=(x_i,y_i,z_i)^t$ for $i=1,\ldots,6$.

$$M(q)=\mathrm{diag}(m_i,\ m_i,\ m_i l_i^3)\quad \text{for } i=1,\ldots,6,$$

$$\begin{aligned}Q(\dot{q},q)=&\,(0,-m_1g,0,0,-m_2g,0,0,-m_3g,T_3,0,-m_4g,0,0,\\&-m_5g,0,0,-m_6g,0)^t,\end{aligned}$$

$$\begin{aligned}\phi(q)\ &(x_1,y_1,\phi_1,x_1-x_2+2\cos\phi_2,y_1-y_2+2\sin\phi_2,x_1-x_3-2\sin\phi_1.\\&y_1-y_3+2\cos\phi_1,x_3-x_4+1.5\cos\phi_3,y_3-y_4+1.5\sin\phi_3.\\&x_5-x_2+0.95\cos\phi_5-2\cos\phi_2,y_5-y_2+0.95\sin\phi_5-2\sin\phi_2,\\&x_5-x_6-0.95\cos\phi_5,\\&y_5-y_6-0.95\sin\phi_5,(x_4-x_2)\sin\phi_2-(y_4-y_2)\cos\phi_2,\\&\sin(\phi_4-\phi_2),(x_6-x_1)\sin\phi_1\\&-(y_6-y_1)\cos\phi_1,\sin(\phi_6-\phi_1))^t.\end{aligned}$$

Table 2
Three link cylindrical coordinate manipulator

	PA	TP	ECS	ICS
$\text{Tol} = 10^{-6}$				
NF	302	378	–	298
NS	151	183	340	150
NJ	308	1034	328	303
$N\phi$	260	1030	1680	303
CPU	6.1	7.2	9.1	3.1
NI	1.72	3.1	2.2	–
NT	822	1370	2218	–
NFK	–	390	–	–
NP	–	4	–	–
$\text{TOL} = 10^{-4}$				
NF	190	210	–	178
NS	95	105	256	90
NJ	195	460	240	183
$N\phi$	155	472	1396	183
CPU	3.8	4.5	6.2	1.9
NI	1.63	2.1	2.5	–
NT	498	698	1548	–
NFK	–	235	–	–
NP	–	6	–	–
$\text{TOL} = 10^{-2}$				
NF	85	110	–	88
NS	43	55	105	45
NJ	91	245	98	93
$N\phi$	79	220	512	93
CPU	2.0	2.5	3.8	1.1
NI	1.8	1.8	2.0	–
NT	242	394	624	–
NFK	–	130	–	–
NP	–	8	–	–

The nonzero elements of the Jacobian matrix $J(q)$ and $v(\dot{q}, q)$ can be computed easily.

Remarks: Dynamic analysis is carried out with a slider mass $m_6 = 50$ kg, a torque $T_3 = 165,521$ Nm applied to flywheel, and flywheel polar Moment of Inertia (MI) is 200 kg-m^2 and with initial velocity zero. The applied torque is selected so that the work done in one cycle of operation ($2\pi T_3$) is equal to the work done in cutting the workpiece. This system is simulated using the four approaches PA, TP, ECS, ICS and the results are tabulated in Table 4.

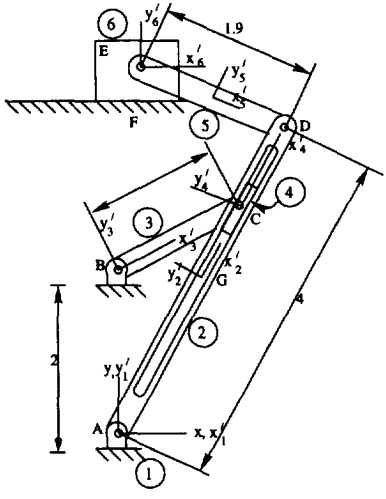

Fig. 3. Quick-return mechanism.

5.1. Observations, recommendations and conclusions

From the tables it appears that ICS is the most efficient approach. By the remark (Remark 5.1) the selection of the stabilizing parameter α in the case of ICS approach is a crucial issue and no clear general way is available. In the examples we have tried, the parameter α is chosen in the following way. We start with $\alpha = 0$ (i.e., integrating just the underlying ODE) and slowly increase its value. By looking at the amount of constraint violation and the number of failed integration steps a proper (optimal) α is chosen. So while actually comparing the performance indicators of this approach with the other approaches it is very important to keep in mind the huge exercise involved in choosing this optimal α. Since the selection of α is highly problem dependent the ICS is not a suitable approach for general purpose use. Therefore in our comparison of the different approaches we exclude it.

Table 3
Inertial properties and CIC

No.	1	2	3	4	5	6
Mass	1	100	1000	5.0	30	60
MI	1	100	2000	0.5	10	1.5
x	0	0.91822	0	1.35993	0.91296	−0.01053
y	0	1.77676	2	2.63293	3.77637	3.99923
ϕ	0	1.90380	0.4356	1.90380	0.23680	0

Table 4
Quick-return mechanism

	PA	TP	ECS	ICS
$TOL = 10^{-6}$				
NF	2160	3112	–	2082
NS	1080	1556	3574	1044
NJ	2060	9512	3787	2087
Nϕ	2054	9482	11232	2087
CPU	77.2	86.3	98.5	52.2
NI	1.98	2.5	2.3	–
NT	6190	12012	18612	–
NFK	–	4312	–	–
NP	–	14	–	–
$TOL = 10^{-4}$				
NF	1272	1876	–	1191
NS	686	938	1932	599
NJ	1173	5724	1860	1196
Nϕ	1166	5430	8632	1196
CPU	45.5	58.9	78.7	29.9
NI	1.98	1.99	2.3	–
NT	3524	9876	12712	–
NFK	–	2842	–	–
NP	–	16	–	–
$TOL = 10^{-2}$				
NF	712	1012	–	676
NS	356	506	1208	346
NJ	682	3814	1194	681
Nϕ	745	3712	5977	681
CPU	24.3	34.8	48.2	17.1
NI	1.9	2.0	2.1	–
NT	1930	6812	8308	–
NFK	–	1864	–	–
NP	–	10	–	–

Based on the data from the numerical examples we make the following observations. (1) In terms of all the performance indicators, PA is the best, in spite of the 1.15 factor [4]. (2) ECS takes substantially more integration steps for all the examples as expected. (3) NJ, Nϕ, and NT are substantially more for TP and ECS and so the effort is more. (4) Finally in the case of TP, the parameter NFK (step 1 of Procedure TP-CMS) reflects the extra effort in the case of TP and is of the order NS. Also, choosing the ratio μ is very important. The number of new parameterizations done depends mainly on this optimal choice of μ. Also as repeatedly mentioned the number of new parameterizations done also depends on ρ, the rate of convergence.

Finally we recommend the following. To start with, solve the given CMS using ICS approach with some $\alpha \neq 0$. If the integration fails repeatedly to accept a step and the step size becomes too small and the minimum step size is reached, and also if the constraint violation is too large, abort this approach. Now start the more accurate and involved approach PA and use it with stiffness detection in the integration routine. The ideas in Refs. [18,19] can be used for this purpose. The code DAMES detects stiffness using these ideas. If stiffness is detected, abort PA and use ECS.

We argue that the PA is better than the Parameterization approach. The chief defects of the Parameterization approach are that: (i) each f evaluation requires the solution of an m-dimensional nonlinear system of equations; and (ii) since the parameterization is local, a change in parameterization may be required during the solution, leading to an integration restart with associated inefficiencies. The PA does not suffer from these defects. It requires only one solution of an m-dimensional nonlinear system of equations in each integration step (step 2 of procedure PA). Also, it does not require any integration restarts because it deals with the full ODE system in Eq. (2.2). The Parameterization approach has the advantage that it integrates only the $(n - m)$-dimensional system of ODEs, whereas the Perturbation approach requires the integration of the n-dimensional system of ODEs. This advantage, however, is only slight because the difference in the integration overhead costs of the two approaches is only $O(m)$ whereas the cost of every extra m-dimensional nonlinear system solution required by the Parameterization approach is $O(m^3)$.

References

[1] E.J. Haug, Computer Aided Kinematics and Dynamics of Mechanical Systems, Basic Methods, Allyn and Bacon, Newton, MA, vol. I, 1989.
[2] W.O. Schiehlen (Ed.), Multibody handbook, Springer Berlin, 1990.
[3] E.J. Haug, Special issue for numerical integration of DAE of mechanical system dynamics, Mech. Structures Mach. 19 (1991).
[4] R. Sudarsan, S. Sathiya Keerthi, Numerical Approaches for solution of differential equations on manifolds, Appl. Math. Comput. 92 (1998) 153–193.
[5] R. Sudarsan, A new approach for the numerical solution of constrained mechanical systems, Ph.D. Thesis, Indian Institute of Science, Bangalore, 1992.
[6] R.A. Wehage, E.J. Haug, Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems, Trans. ASME 104 (1982) 247–255.
[7] N.K. Mani, E.J. Haug, K.E. Atkinson, Application of singular value decomposition for analysis of mechanical system dynamics, Trans. ASME 107 (1985) 82–87.
[8] J. Baumgarte, Stabilization of constraints and integrals of motion in dynamical systems, Comput. Methods Appl. Mech. Eng. 1 (1972) 1–16.
[9] C.W. Gear, B. Leimkuhler, G.K. Gupta, Automatic integration of Euler–Lagrange equations with constraints, J. Comput. Appl. Math. 12 (1985) 77–90.
[10] C. Führer, B. Leimkuhler, Formulation and numerical solution of the equations for constrained mechanical motion, Technical Report FB-08, DFVLR, Koeln, Germany, 1989.

[11] W.C. Rheinboldt, Differential-algebraic systems as differential equations on manifolds, Math. Comp. 43 (1984) 473–482.
[12] P.E. Nikravesh, Computer-Aided analysis of mechanical systems, Prentice Hall, Englewood Cliffs, NJ, 1988.
[13] S.L. Campbell, B. Leimkuhler, Differentiation of constraints in differential-algebraic equations, Mechanics Structures Machines 19 (1991) 19–40.
[14] K.R. Jackson, R.S. Davis, An alternative implementation of variable stepsize multistep formulas for stiff ODEs, A.C.M. Trans. Math. Soft. 6 (1980) 295–318.
[15] L.R. Petzold, A description of DASSL: A differential/algebraic system solver, in: R.S. Stepleman et al. (Ed.), Scientific Computing, North-Holland, Amsterdam, 1983, pp. 65–68.
[16] L.F. Shampine, Implementation of implicit formulas for the solution of ODEs, SIAM J. Sci. Statist. Comput. 1 (1980) 103–118.
[17] L.F. Shampine, M.K. Gordon, Computer solution of Ordinary Differential Equations, Freeman, New York, 1975.
[18] H. Krishnan, N.H. McClamroch, A new approach to position and contact free regulation in constrained robot systems, Proc. IEEE Int. Conf. on Robotics and Automation, Cincinnati, 1990.
[19] L.F. Shampine, Stiffness and nonstiff differential equation solvers II: Detecting stiffness with Runge-Kutta methods, ACM Trans. Math. Soft. 3 (1977) 44–53.
[20] L.F. Shampine, K.L. Hiebert, Detecting Stiffness with Fehlberg (4,5) formulas, Comput. Math. Appl. 3 (1977) 41–46.